UNITED STATES PATENT APPLICATION

System, Method, and Software for Estimation of Motion Vectors

INVENTOR

ANDRE ZACCARIN
of Sunnyvale, California USA

Eduardo E. Drake
Schwegman, Lundberg, Woessner, & Kluth, P.A.
1600 TCF Tower
121 South Eighth Street
Minneapolis, Minnesota 55402

ATTORNEY DOCKET 884.447US1
Client Reference P11222

System, Method, and Software for Estimation of Motion Vectors

Technical Field

The present invention concerns systems and methods for storing and transmitting sequences of digital images, particularly systems and methods for rapid computation of motion vectors.

Background

In recent years, it has become increasingly common to communicate digital video information ---sequences of digital images--- from one point to another, particularly over computer networks, such as the World-Wide-Web portion of the Internet. Since a single frame of video can consists of thousands or even hundreds of thousands of bits of information, it can take a considerable amount of time to transmit a sequence of frames from one point to another.

To reduce transmission times and conserve storage space, computers and other devices that use digital video data often include a video compression system. The video compression system typically includes an encoder for compressing digital video data and a decoder for decompressing, or reconstructing, the digital video data from its compressed form.

Video compression typically takes advantage of the redundancy within and between sequential frames of video data to reduce the amount of data ultimately needed to represent the video data. For example, in a one-minute sequence of frames showing a blue stationwagon passing through an intersection of two neighborhood streets, the first 75 percent of the frames in the sequence may only show the intersection itself, nearby houses, and parked cars, and the remaining 25 percent may show the blue stationwagon moving through the intersection. In this case, 75 percent of the frames could be compressed to a single frame plus information about how many times to repeat this frame before showing the frames with the blue stationwagon.

However, even the frames with the blue stationwagon can be compressed given that the background of nearby houses and parked cars remains essentially constant from frame to frame as the stationwagon moves through the intersection. Indeed, conventional video compression techniques would compress the frames showing the blue stationwagon to a set of image data for the blue stationwagon and data indicating position of the stationwagon relative to other portions

of the background, such as the streets, houses, and parked cars. The information about relative position of the stationwagon from one frame to the next is generally called a motion or displacement vector.

In general, computing motion vectors is computationally intensive, since unlike the simple example of the blue stationwagon, a video encoder must determine for itself what is redundant or reusable from one frame to the next. Many, if not most, systems determine the motion vectors using a block-matching algorithm.

Block matching entails dividing a given frame into blocks of pixels, and for each block, searching a designated area of the previous frame for the block of pixels that is most similar to it, based on a performance criterion. The location of this "best matching" block relative to the block in the given frame defines a motion vector for the given block. This means that the encoder can represent this block as the location of the "best matching" block from the previously sent frame plus any differences between pixels in the best matching block and those in the block being compressed. Note that if a block in the current frame and a block in the previous frame are identical, such as two blocks that represent the door of a blue station wagon, all differences will be zero and the block in the current frame can be encoded as a coordinate vector identifying the location of the corresponding block in the previous frame plus a code indicating that all differences are zero.

Most of the work in determining a motion vector occurs in comparing each block in the frame being compressed to blocks within the search area of the reference frame. There are numerous ways of comparing one block to another. One common way entails computing the sum of absolute differences between each pixel in the block being encoded and a corresponding pixel in a block of pixels from the search area.

Although the search areas are maybe relatively small compared to the frame size, the number of possible matching blocks within the search area and the use of all the pixels in each of these blocks still requires a significant amount of time to determine a motion vector.

Accordingly, there is a continuing need for faster methods of computing motion vectors.

Brief Description of the Drawings

Figure 1 is a block diagram of a computer system 100 incorporating teachings of the present invention.

Figure 2 is a flow chart of an exemplary method incorporating teachings of the present invention.

Figure 3 is a diagram showing a target frame 310, a reference frame 320, and a search area 322.

Detailed Description of Exemplary Embodiments

The following detailed description, which references and incorporates the above-identified figures, describes and illustrates one or more specific embodiments of the invention. These embodiments, offered not to limit but only to exemplify and teach, are shown and described in sufficient detail to enable those skilled in the art to implement or practice the invention. Thus, where appropriate to avoid obscuring the invention, the description may omit certain information known to those of skill in the art.

Figure 1 shows an exemplary video compression system 100. Exemplary system 100 includes one or more processors 110, memory 120, video or image decoder 130, and video or image encoder 140 intercoupled via a wireline or wireless bus 150. (Decoder 130 and encoder 140 are shown as broken-lines boxes to emphasize that they may exist as hardware or software devices.) Exemplary processors include Intel Pentium processors; exemplary memory includes electronic, magnetic, and optical memories; and exemplary busses include ISA, PCI, and NUBUS busses. (Intel and Pentium are trademarks of Intel Corporation, and NUBUS is a trademark of Apple Computer.)

Of particular interest, video encoder 140 includes and a motion estimation module 142. Various embodiments implement module 142 as a set of computer-executable instructions, an application-specific integrated circuit, or as a combination of computer-executable instructions and hardware. (In some embodiments, video encoder 140 includes a separate processor.) Indeed, the scope of the present invention is believed to encompass software, hardware, and firmware implementations.

In general operation, video encoder 140 receives a sequence of video images, or frames, and encodes or compresses them according to one or more intraframe and/or interframe video encoding or compression standards, such as Moving Pictures Experts Group 1, 2, or 4 (MPEG-1, MPEG-2, or MPEG-4), or International Telecommunication Union H.261, H63, or H.263+ Videoconferencing Standards. As part of the otherwise conventional encoding process, motion-estimation module 142 estimates motion vectors for a target block of pixels by subsampling blocks in a search area of a reference frame of video data, measuring distortion based on a subsampling of pixels from the blocks, and using the block with minimum distortion to estimate a motion vector for the target block. The motion vector is then used to encode the target block.

More particularly, Figure 2 shows a flow chart 200 that illustrates an exemplary method of operating video encoder 140, including a method of estimating motion vectors. Flow chart 200 includes blocks 210-270, which are arranged serially in the exemplary embodiment. However, other embodiments of the invention may execute two or more blocks in parallel using multiple processors or a single processor organized as two or more virtual machines or subprocessors. Moreover, still other embodiments implement the blocks as two or more specific interconnected hardware modules with related control and data signals communicated between and through the modules, or as portions of an application-specific integrated circuit. Thus, the exemplary process flow is applicable to software, firmware, and hardware implementations.

Block 210 entails receiving or retrieving an M-by-N reference frame or field F_r and an M-by-N target frame or field F_t of a video sequence, or a subsampled version of the frame or field. Frames F_r and F_t respectively comprise a number of reference blocks $B_r(x,y)$ and target blocks $B_t(x,y)$, each of which includes an m-by-n (m columns x n lines or rows) array of pixels, with the upper left pixel in the block having the coordinates (x, y). (All blocks in this description are assumed to be rectangular and are identified based on their upper left most pixel coordinates; however, the invention is not limited to any block shape or particular convention for defining blocks.)

In the exemplary embodiment, reference frame or field F_r precedes or succeeds target frame F_t in a video or image sequence by one or more frames. However, in other embodiments, for example, some that employ intra-frame encoding, reference frame or field is contained within

target frame F_r. Exemplary execution continues at block 220.

Block 220 entails identifying a target block $B_t(x_0, y_0)$ from target frame F_t and defining a corresponding search area within reference frame F_r. The target block is the block that the video encoder will encode. In some embodiment, two or more target blocks and corresponding search areas are selected and defined to facilitate parallel encoding of the target blocks.

Although the present invention is not limited to any particular target-block identification or search-area definition, the exemplary embodiment centers the search area around coordinates in the reference frame that correspond to or approximate center coordinates of the target block within the target frame. However, other embodiments center the search area on coordinates that are likely to correspond to the coordinates of the best matching block, as determined , for example, by the motion vectors of neighboring blocks. Additionally, the exemplary embodiment defines the search area to smaller than the reference frame and larger than the target block.

Figure 3 illustrates a target frame 310 and a reference frame 320. Target frame 310 includes a target block 312, and reference frame 320 includes a search area 322. In the exemplary embodiment, the search area is 15x15 or 31x31 pixels; however, the invention is not limited to these search-area dimensions. The exemplary embodiment defines the search area as the set of upper-left coordinate pixels that define a set of corresponding blocks. However, some other embodiments define the search area in terms of the total set of pixels considered when looking for matching blocks. After identifying one or more target blocks and corresponding search areas, execution proceeds to block 230.

Block 230 entails determining K candidate blocks from reference frame F_r that minimizes a partial distortion measure relative to the selected target block $B_t(x_0, y_0)$, with the partial distortion measure based on a predetermined set of pixels in both blocks. If there is a tie among two or more blocks, the first candidate block that yielded the minimum is selected; however, other embodiments may break the tie using other methods, such as minimization of encoding cost.

More precisely, each *k*-th candidate block in the reference frame is denoted $B_r^*(a_k^*, b_k^*)$, where the *k*-th coordinate pair (a_k^*, b_k^*), or candidate motion vector, is defined as

$$(a_k^*, b_k^*) = \arg\min\left[D_{l(k)}(a,b) \; for\,(a,b) \in S_k\right] \; for\; k = 1..K$$

$D_{l(k)}(a,b)$ denotes a partial-distortion measure based on a *k*-th set of pixels *l(k)* within the block $B_r(a,b)$ of the reference frame, and S_k denotes a *k*-th predetermined set of coordinate pairs that defines a particular set of candidate blocks within the search area of the reference frame. Arg min[:] denotes the argument that minimizes the bracketed quantity. In this case, it means the coordinate pair (a,b) within S_k that yields the lowest partial-distortion measure.

In the exemplary embodiment, K is 16, and *l(k)* is defined as the *k*-th line (or column) of pixels in a given block. Thus, the exemplary embodiment defines 16 mutually exclusive subsampling patterns *l(1), l(2), ..., l(16)*. However, other embodiments define *l(k)* as every other pixel in the *k*-th line, as two or more complete or partial lines within a block. And, still other embodiments define *l(k)* as a subset of non-collinear pixels within the block.

The exemplary embodiment also defines each set of coordinates S_k to contains the coordinates for every other pixel in each *k*-th column or row of the search area. For example, if the search area is 17x17 and the block size is 16x16, S_1 would contain coordinates identifying every other pixel in the first and seventeenth (17 mod 16 = 1) columns of the search area, and S_2 would contain coordinates identifying every other pixel in the second column. To further illustrate, Figure 3 shows a search area with each pixel labeled 1, 2, 3, ... 16, indicating its respective association with coordinate sets S_1, S_2, S_3, ..., S_{16}. Alternatively, for an N-column search area and KxK blocks, one can determine the columns for S_i as i, i+K, i+2K, i+3K, and so forth, or as i+nK, for all n $\geq$0 such that i+nK $\leq$N.

Other embodiments use other sizes and shapes of blocks and different levels of search-area subsampling. For example, one embodiment uses a 32x32 pixel search area and defines S_k to include every pixel or every fourth, eighth, or sixteenth pixel from each k-th column of the

search area.

The exemplary embodiment computes $D_{l(k)}(a,b)$ as the Sum of Absolute Differences. More precisely, $D_{l(k)}(a,b)$ is defined as

$$D_{l(k)}(a,b) = \sum_{(i,j)\in l(k)} \left| (B_t(x+i, y+j) - B_r(x-a+i, y-b+j) \right|$$

However, other embodiments use other distortion-measurement or matching criterion, such as mean absolute difference (MAD) or mean squared error (MSE). Thus, the present invention is believed not to be limited to any particular species or genus of distortion measurement.

The exemplary embodiment uses SIMD (single-instruction-multiple-data) MMX or SSE type instructions, such as the PSAD instruction in the SSE2 instruction set for the Intel Pentium 4 microprocessor, to compute this distortion measure. (Intel and Pentium are trademarks of Intel Corporation.) Use of this type of instruction allows parallel computation of the distortion functions.

Block 240, which is executed after determining the set of K candidate blocks (and associated coordinate vectors) in block 230, entails selecting the vector associated with the block $B_r^*(a_k^*, b_k^*)$ that minimizes a distortion measure *D(a,b)*. In other words,

$$(a^*, b^*) = \arg\min D(a,b) \; for \; (a,b) \in \left\{ (a_k^*, b_k^*),\, k = 1, \ldots K \right\}$$

where *D(a,b)* is defined as

$$D(a,b) = \sum_{j=1}^{m} \sum_{i=1}^{n} \left| (B_t(x_0 + i, y_0 + j) - B_r(x_0 - a + i, y_0 - b + j) \right|$$

If more than one block yields the same minimum distortion, there are a number of ways to resolve the tie. For example, the block having the lowest cost of encoding can be selected.

Rather than compute another set of distortion measures based on *D*, some embodiments simply select the coordinate vector (a_k^*, b_k^*) associated with candidate block $B_r^*(a_k^*, b_k^*)$ that yielded the lowest partial-distortion measurement $D_{l(k)}(a,b)$. In mathematical terms, this is

expressed as

$$(a^*, b^*) = \arg\min[D_{l(k)}(a_k^*, b_k^*) \; for \; k = 1..K]$$

Again, if there are multiple minima, the exemplary embodiment selects the block that has the lowest encoding cost.

At block 250, after selecting the one of the candidate vectors, the exemplary embodiment encodes block B_t of frame F_t. This entails computing the motion vector for the target block as

$$V(x_o, y_o) = (a^*, b^*)$$

and a difference matrix DM as

$$DM = B_t(x_0, y_0) - B_r(x_0 - a^*, y_0 - b^*)$$

The exemplary embodiment uses this motion vector V and difference matrix DM to encode the target block, specifically forming packets of digital data according to MPEG-1, 2, 4, H.261, H263, H.263+, and/or other suitable protocols.

In decision block 260, the exemplary method determines if the target frame is completely encoded. If it is not fully encoded, meaning that there are additional blocks of the target frame that require encoding, execution returns to process block 220 to initiate selection and encoding of another target block from the target frame. However, if the target frame is fully encoded, execution proceeds to process block 270.

Block 270 entails outputting the packets of encoded data representative of the target frame. The exemplary embodiments outputs the data to a memory for storage and/or transmission to remote display device.

Conclusion

In furtherance of the art, the present inventor has presented methods, systems, and software for rapid estimation of motion vectors.

The embodiments described above are intended only to illustrate and teach one or more ways of practicing or implementing the present invention, not to restrict its breadth or scope. The actual scope of the invention, which embraces all ways of practicing or implementing the teachings of the invention, is defined only by the following claims and their equivalents.